

# PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2000-207200

(43)Date of publication of application : 28.07.2000

(51)Int.Cl.

G06F 9/06  
G06F 12/14

(21)Application number : 11-361095

(71)Applicant : NETACTIVE INC

(22)Date of filing : 20.12.1999

(72)Inventor : GORDON EDWARD RAROOZU  
SCOTT ALAN TOMSON

(30)Priority

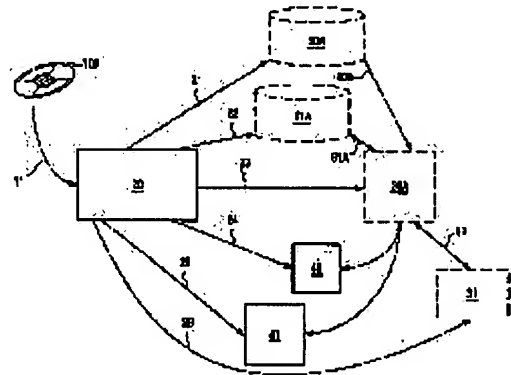
Priority number : 98 218413 Priority date : 22.12.1998 Priority country : US

## (54) SOFTWARE PROGRAM PROTECT SYSTEM AND MACHINE-READABLE MEDIUM

(57)Abstract:

**PROBLEM TO BE SOLVED:** To prevent revision and redistribution of unapproved computer software by reconverting the converted data inversely by using a conversion key and sending the data in an unconverted state to an execution program.

**SOLUTION:** Converting file sets 50A and 51A are accessed by a corrected execution program 30A via an interface reading/writing function 61A and an interface read-only function 60A. These functions provide proper run-time conversion and inverse conversion. When there is file operation hindrance, a file converting module program retrieves converted data from at least one converted data file, converts the converted data inversely into its unconverted state by using a conversion key, and sends the data in the unconverted state to an executing program.



## LEGAL STATUS

[Date of request for examination]

23.01.2001

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11)特許出願公開番号  
特開2000-207200  
(P2000-207200A)

(43)公開日 平成12年7月28日(2000.7.28)

(51) Int.Cl.	識別記号	F I	ページ(参考)
G 0 6 F 9/06	5 5 0	G 0 6 F 9/06	5 5 0 Z
12/14	3 2 0	12/14	3 2 0 F

審査請求 未請求 請求項の数46 O.L (全 16 頁)

(21)出願番号	特願平11-361095	(71)出願人	399126743 ネットアクティブ・アイエヌシー NETACTIVE INC. カナダ国, ケイ2イー 8シー4 オンタ リオ, ネビーン, スイート 530, アンテ アーズ ドライブ 1
(22)出願日	平成11年12月20日(1999. 12. 20)	(72)発明者	ゴードン・エドワード・ラローズ カナダ国, ケイ2シー 0イー3, オンタ リオ, オタワ, ベースライン ロード 2417
(31)優先権主張番号	09/218413	(74)代理人	100097216 弁理士 泉 和人 (外1名)
(32)優先日	平成10年12月22日(1998. 12. 22)		
(33)優先権主張国	米国(US)		

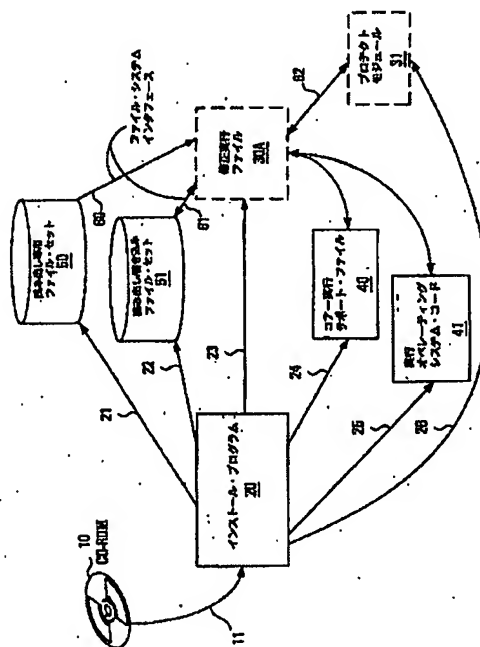
最終頁に続く

(54)【発明の名称】 ソフトウェア・プログラム・プロテクト・システムおよび機械読み出しが可能な媒体

(57) 【要約】

【課題】 本発明は、非認可のコンピュータ・ソフトウェア・プログラムの修正・再配布を防止するプロテクト機構を提供する。

【解決手段】 本発明は、ソフトウェア・プログラムのアセット・ファイルとコア実行プログラム自身の両方を変換するソフトウェア・プロテクト機構を開示する。そこでは、修正された実行プログラムと対応の変換アセット・ファイルを再配布することが必要である。ファイル変換モジュールは、オペレーティング・システムからの全てのファイル・アクティビティを妨害するために使用され、またオペレーティング・システムへの要求が修正実行プログラムへの（からの）変換データに対して起動されたと識別されたときに限り逆変換を実行する。アセット・リスト、呼プロセス識別情報、変換キー、および変換アルゴリズム等を含むプロテクト・モジュールは、ファイル変換モジュールと関連して使用される。



## 【特許請求の範囲】

【請求項1】 (a) 少なくとも1つの変換データ・ファイルをストアするメモリと、(b) オペレーティング・システム、実行プログラム、および前記の実行プログラムに変換サービスを提供するファイル変換モジュール・プログラムを実行するプロセッサを有し、

前記ファイル変換モジュール・プログラムは、前記実行プログラムから前記オペレーティング・システムへのファイル・オペレーションを妨害し、

ここで、ファイル・オペレーションの妨害時に、前記ファイル変換モジュール・プログラムは、少なくとも1つの前記変換データ・ファイルから変換データを検索し、変換キーを用いて、前記変換データを、その未変換状態に逆変換し、前記データを、未変換状態のまま、前記実行プログラムに送ることを特徴とするソフトウェア・プログラム・プロテクト・システム。

【請求項2】 請求項1記載のシステムにおいて、前記ファイル・オペレーションは、ファイル・オープン・ファイル・オペレーションと、ファイル読み出しファイル・オペレーションと、ファイル書き込みファイル・オペレーションと、ファイル・クローズ・ファイル・オペレーションとを含むことを特徴とするソフトウェア・プログラム・プロテクト・システム。

【請求項3】 請求項2記載のシステムにおいて、ファイル・オープン・ファイル・オペレーションが行われる際、前記ファイル変換モジュール・プログラムは、少なくとも1つの前記変換データ・ファイルに対して、独自のファイル識別子を割り当て、前記独自のファイル識別子を、アクティブ・ファイル・リスト中に記録することを特徴とするソフトウェア・プログラム・プロテクト・システム。

【請求項4】 請求項3記載のシステムにおいて、ファイル読み出しおよびファイル書き込みファイル・オペレーションを行う際、前記ファイル変換モジュール・プログラムは、アクティブ・ファイル・リストをチェックして、逆変換を行うべきかどうかを決定し、行わない場合には、前記ファイル・オペレーションを、通常処理用の前記オペレーティング・システムに戻すことを特徴とするソフトウェア・プログラム・プロテクト・システム。

【請求項5】 請求項3記載のシステムにおいて、前記変換データ・ファイルを閉じる際、前記ファイル変換モジュール・プログラムは、前記アクティブ・ファイル・リストから、少なくとも1つの変換データ・ファイルに対する前記独自のファイル識別子を検出することを特徴とするソフトウェア・プログラム・プロテクト・システム。

【請求項6】 請求項1記載のシステムにおいて、少なくとも1つの前記変換データ・ファイルは、読み出し専用ファイルであることを特徴とするソフトウェア・プログラム・プロテクト・システム。

【請求項7】 請求項1記載のシステムにおいて、少なくとも1つの前記変換データ・ファイルは、読み出し/書き込みファイルであることを特徴とするソフトウェア・プログラム・プロテクト・システム。

【請求項8】 請求項1記載のシステムにおいて、前記実行プログラムは、前記実行プログラムが実行されるときにだけ起動されるプロテクト・モジュール・プログラムを実行するための呼と共に組み込まれ、前記プロテクト・モジュール・プログラムは、前記実行プログラムを最初に実行するとき、前記ファイル変換モジュールを初期化することを特徴とするソフトウェア・プログラム・プロテクト・システム。

【請求項9】 請求項8記載のシステムにおいて、前記変換キーは、前記プロテクト・モジュール・プログラム内にストアされることを特徴とするソフトウェア・プログラム・プロテクト・システム。

【請求項10】 請求項8記載のシステムにおいて、前記プロテクト・モジュール・プログラムは、ダイナミック・リンク・ライブラリ(DLL)であることを特徴とするソフトウェア・プログラム・プロテクト・システム。

【請求項11】 請求項1記載のシステムにおいて、少なくとも1つの前記変換データ・ファイルは、DESおよびRSA暗号化アルゴリズムのうちの1つを用いて暗号化されることを特徴とするソフトウェア・プログラム・プロテクト・システム。

【請求項12】 請求項1記載のシステムにおいて、非実行ライセンス・データを前記実行プログラム中に組み込み、前記ファイル変換モジュールが変換サービスを前記実行プログラムに確実に供給することを特徴とするソフトウェア・プログラム・プロテクト・システム。

【請求項13】 請求項1記載のシステムにおいて、実行プログラム・コードを前記実行プログラム中に組み込み、前記ファイル変換モジュールが変換サービスを前記実行プログラムに確実に供給することを特徴とするソフトウェア・プログラム・プロテクト・システム。

【請求項14】 実行プログラムと1以上のデータ・ファイルを含むコンピュータ・ソフトウェア・アプリケーションから、コンピュータ・ソフトウェア・アプリケーションのプロテクト版を生成する方法において：

(i) 1以上の変換アルゴリズムと変換キーを用いて、1以上のデータ・ファイルを、1以上の各変換データ・ファイルに変換し、

(i i) 前記変換キーをストアし、

(i i i) 変換サービスを前記実行プログラムに供給するファイル変換モジュール・プログラムを検索し、

(i v) 前記の1以上の各変換データ・ファイルと、変換キーと、実行プログラムと、ファイル変換モジュール・プログラムとを、ユーザに配布するための媒体に転送するステップからなることを特徴とするコンピュータ

10

20

30

40

50

・ソフトウェア・アプリケーションのプロテクト版生成方法。

【請求項15】 請求項14記載の方法において、前記実行プログラムが実行されるときにだけ起動されるプロテクト・モジュール・プログラムを実行するための呼を前記実行プログラムに組み込むことによって前記実行プログラムを修正するステップをさらに含み、前記プロテクト・モジュール・プログラムは、前記実行プログラムを最初に実行するとき、前記ファイル変換モジュールを初期化することを特徴とするコンピュータ・ソフトウェア・アプリケーションのプロテクト版生成方法。

【請求項16】 請求項15記載の方法において、前記変換キーをストアするステップは、前記プロテクト・モジュール・プログラム内に前記変換キーを組み込むステップを含むことを特徴とするコンピュータ・ソフトウェア・アプリケーションのプロテクト版生成方法。

【請求項17】 請求項15記載の方法において、前記プロテクト・モジュール・プログラムは、ダイナミック・リンク・ライブラリー（DLL）であることを特徴とするコンピュータ・ソフトウェア・アプリケーションのプロテクト版生成方法。

【請求項18】 請求項14記載の方法において、前記変換アルゴリズムは、DESおよびRSA暗号化アルゴリズムのうちの1つから選択されることを特徴とするコンピュータ・ソフトウェア・アプリケーションのプロテクト版生成方法。

【請求項19】 請求項14記載の方法において、非実行ライセンス・データを前記実行プログラム中に組み込むステップをさらに含み、前記ファイル変換モジュールが変換サービスを前記実行プログラムに確実に供給することを特徴とするコンピュータ・ソフトウェア・アプリケーションのプロテクト版生成方法。

【請求項20】 請求項14記載の方法において、実行プログラム・コードを前記実行プログラム中に組み込むステップをさらに含み、前記ファイル変換モジュールが変換サービスを前記実行プログラムに確実に供給することを特徴とするコンピュータ・ソフトウェア・アプリケーションのプロテクト版生成方法。

【請求項21】 少なくとも1つの変換データ・ファイルと、少なくとも1つの変換キーと、変換サービスを前記実行プログラムに供給するファイル変換モジュール・プログラムとを含む機械読み出しが可能な媒体において、

前記実行プログラムと前記ファイル変換モジュール・プログラムとがコンピュータ・システム上で走るとき、前記ファイル変換モジュール・プログラムは、前記実行プログラムから前記コンピュータ・システムのオペレーティング・システムへのファイル・オペレーションを妨害し、

ここで、ファイル・オペレーションの妨害時に、前記ファイル変換モジュール・プログラムは、前記の少なくとも1つの変換データ・ファイルから変換データを検索し、前記の少なくとも1つの変換キーを使用して、前記変換データをその未変換状態に逆変換し、前記データを未変換状態で前記実行プログラムに送ることを特徴とする機械読み出しが可能な媒体。

【請求項22】 請求項21記載の媒体において、前記ファイル・オペレーションは、ファイル・オープン・ファイル・オペレーションと、ファイル読み出しファイル・オペレーションと、ファイル書き込みファイル・オペレーションと、ファイル・クローズ・ファイル・オペレーションとを含むことを特徴とする機械読み出しが可能な媒体。

【請求項23】 請求項22記載の媒体において、ファイル・オープン・ファイル・オペレーションを妨害するとき、前記ファイル変換モジュール・プログラムは、前記の少なくとも1つの変換データ・ファイルに対して独自のファイル識別子を割り当て、前記独自のファイル識別子を、アクティブ・ファイル・リスト中に記録することを特徴とする機械読み出しが可能な媒体。

【請求項24】 請求項23記載の媒体において、ファイル読み出しファイル・オペレーションまたはファイル書き込みファイル・オペレーションを妨害するとき、ファイル変換モジュール・プログラムは、アクティブ・ファイル・リストをチェックして、逆変換を行うべきかどうかを判断し、行わない場合には、前記ファイル・オペレーションを、通常処理用の前記オペレーティング・システムに戻すことを特徴とする機械読み出しが可能な媒体。

【請求項25】 請求項23記載の媒体において、コンピュータ・システムが走っているとき、ファイル変換モジュール・プログラムは、前記の少なくとも1つの変換データ・ファイルを閉じるときに、前記アクティブ・ファイル・リストから、前記の少なくとも1つの変換データ・ファイルに対する独自のファイル識別子を削除することを特徴とする機械読み出しが可能な媒体。

【請求項26】 請求項21記載の媒体において、前記の少なくとも1つの変換データ・ファイルは、読み出し専用ファイルであることを特徴とする機械読み出しが可能な媒体。

【請求項27】 請求項21記載の媒体において、前記の少なくとも1つの変換データ・ファイルは読み出し/書き込みファイルであることを特徴とする機械読み出しが可能な媒体。

【請求項28】 請求項21記載の媒体において、前記実行プログラムは、前記実行プログラムが実行されるときにだけ起動されるプロテクト・モジュール・プログラムを実行するための呼と共に組み込まれ、前記プロテクト・モジュール・プログラムは、前記実行プログラムを

最初に実行するとき、前記ファイル変換モジュールを初期化することを特徴とする機械読み出しが可能な媒体。

【請求項29】 請求項28記載の媒体において、少なくとも1つの変換キーは、前記プロテクト・モジュール・プログラム内に組み込まれることを特徴とする機械読み出しが可能な媒体。

【請求項30】 請求項28記載の媒体において、前記プロテクト・モジュール・プログラムは、ダイナミック・リンク・ライブラリ(DLL)であることを特徴とする機械読み出しが可能な媒体。

【請求項31】 請求項21記載の媒体において、少なくとも1つの前記変換データ・ファイルは、DESおよびRSA暗号化アルゴリズムのうちの1つを用いて暗号化されることを特徴とする機械読み出しが可能な媒体。

【請求項32】 請求項21記載の媒体において、非実行ライセンス・データを前記実行プログラム中に組み込み、前記ファイル変換モジュールが変換サービスを前記実行プログラムに確実に供給することを特徴とする機械読み出しが可能な媒体。

【請求項33】 請求項21記載の媒体において、実行プログラム・コードを前記実行プログラム中に組み込み、前記ファイル変換モジュールが変換サービスを前記実行プログラムに確実に供給することを特徴とする機械読み出しが可能な媒体。

【請求項34】 (a) 少なくとも1つの変換データ・ファイルをストアするメモリと、(b) オペレーティング・システム、修正実行プログラムを実行するプロセッサとを有し、

前記修正実行プログラムは、変換サービスを前記修正実行プログラムに供給するための変換モジュール・プログラムを起動するために修正され、前記ファイル変換モジュール・プログラムは、前記修正実行プログラムから前記オペレーティング・システムへのファイル・オペレーションを妨害し、

ここで、ファイル・オペレーションを妨害する際に、前記ファイル変換モジュール・プログラムは、少なくとも1つの変換データ・ファイルから変換データを検索し、変換キーを用いて、前記変換データを、その未変換状態に逆変換し、前記のデータを、未変換状態のまま、前記の修正実行プログラムに送ることを特徴とするソフトウェア・プログラム・プロテクト・システム。

【請求項35】 請求項34記載のシステムにおいて、前記ファイル・オペレーションは、ファイル・オープン・ファイル・オペレーションと、ファイル読み出しファイル・オペレーションと、ファイル書き込みファイル・オペレーションと、ファイル・クローズ・ファイル・オペレーションとを含むことを特徴とするソフトウェア・プログラム・プロテクト・システム。

【請求項36】 請求項35記載のシステムにおいて、ファイル・オープン・ファイル・オペレーションが行わ

れる際、前記ファイル変換モジュール・プログラムは、少なくとも1つの前記変換データ・ファイルに対して、独自のファイル識別子を割り当て、前記独自のファイル識別子を、アクティブ・ファイル・リスト中に記録することを特徴とするソフトウェア・プログラム・プロテクト・システム。

【請求項37】 請求項36記載のシステムにおいて、ファイル読み出しおよびファイル書き込みファイル・オペレーションを行う際、前記ファイル変換モジュール・プログラムは、アクティブ・ファイル・リストをチェックして、逆変換を行うべきかどうかを決定し、行わない場合には、前記ファイル・オペレーションを、通常処理用の前記オペレーティング・システムに戻すことを特徴とするソフトウェア・プログラム・プロテクト・システム。

【請求項38】 請求項36記載のシステムにおいて、前記変換データ・ファイルを閉じる際、前記ファイル変換モジュール・プログラムは、前記アクティブ・ファイル・リストから、少なくとも1つの変換データ・ファイルに対する前記独自のファイル識別子を検出することを特徴とするソフトウェア・プログラム・プロテクト・システム。

【請求項39】 請求項34記載のシステムにおいて、少なくとも1つの前記変換データ・ファイルは、読み出し専用ファイルであることを特徴とするソフトウェア・プログラム・プロテクト・システム。

【請求項40】 請求項34記載のシステムにおいて、少なくとも1つの前記変換データ・ファイルは、読み出し/書き込みファイルであることを特徴とするソフトウェア・プログラム・プロテクト・システム。

【請求項41】 請求項34記載のシステムにおいて、前記修正実行プログラムは、前記修正実行プログラムが実行されるときにだけ起動されるプロテクト・モジュール・プログラムを実行するための呼と共に組み込まれ、前記プロテクト・モジュール・プログラムは、前記修正実行プログラムを最初に実行するとき、前記ファイル変換モジュールを初期化することを特徴とするソフトウェア・プログラム・プロテクト・システム。

【請求項42】 請求項41記載のシステムにおいて、前記変換キーは、前記プロテクト・モジュール・プログラム内にストアされることを特徴とするソフトウェア・プログラム・プロテクト・システム。

【請求項43】 請求項41記載のシステムにおいて、前記プロテクト・モジュール・プログラムは、ダイナミック・リンク・ライブラリ(DLL)であることを特徴とするソフトウェア・プログラム・プロテクト・システム。

【請求項44】 請求項34記載のシステムにおいて、少なくとも1つの前記変換データ・ファイルは、DESおよびRSA暗号化アルゴリズムのうちの1つを用いて

暗号化されることを特徴とするソフトウェア・プログラム・プロテクト・システム。

【請求項45】 請求項34記載のシステムにおいて、非実行ライセンス・データを前記修正実行プログラム中に組み込み、前記ファイル変換モジュールが変換サービスを前記修正実行プログラムに確実に供給することを特徴とするソフトウェア・プログラム・プロテクト・システム。

【請求項46】 請求項34記載のシステムにおいて、実行プログラム・コードを前記修正実行プログラム中に組み込み、前記ファイル変換モジュールが変換サービスを前記修正実行プログラムに確実に供給することを特徴とするソフトウェア・プログラム・プロテクト・システム。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、非認可のコンピュータ・ソフトウェア・プログラムの修正および/または再配布を防止するプロテクト機構に関するものである。

【0002】

【従来の技術】商業分野のソフトウェア、特に、消費者向け個人コンピュータすなわちPCのソフトウェアの分野では、市場を拡大する目的で、ソフトウェアのデモンストレーション版を、無料で、または、廉価で配布することが一般的に行われている。ソフトウェア販売業者は、自分たちのソフトウェアを、多くの潜在的な購買者に見せるために、このようなソフトウェアのデモンストレーション版を多量に配布している。このようなことが可能なのは、低コストで配布しても、ソフトウェアを試用した結果、これを気に入って購買するユーザがいるために、十分に利益が上がるからである。このようなデモンストレーション版は、よくインターネットを介して配布されるが、一般的には、フロッピーディスクやCD-ROMまたはDVD（多目的光ディスク）のような物理的な媒体の方がよく使用されている。このような物理的な媒体は廉価で、特定の対象者に簡単に届けることができる。また、このような物理的な媒体は、多くの消費者向けインターネット接続を介するダウンロード支援ソフトウェアとは異なり、販売用と同じく完全な機能を持つ大容量のデモンストレーション版を配布するために必要な十分な容量を有している。

【0003】このようなデモンストレーション・ソフトウェア・プログラムは、以下の特性を持つ。すなわち、

(i) 配布形式は、ソフトウェア・プログラムの完全機能版である、(ii) デモンストレーション・ソフトウェア・プログラムの配布されたコピーはすべて同一である、(iii) 通常のインストールでは、このソフトウェア・プログラムは、デモンストレーション仕様を供給し、1つまたはそれ以上の制限（機能性、時間、ユーザの数など）がある、(iv) ユーザがこのソフトウェア

の完全なライセンス版を取得できる手段が提供され、また、ユーザは、追加の媒体の配布を受けることなく、このデモンストレーション版を、完全版に変換できる手段が提供される、(v) このデモンストレーション・ソフトウェア・プログラムは、特定のハードウェアに依存せず、通常の消費者向けPC上で走る（また、完全機能版に切り替わる）。

【0004】概して、デモンストレーション用に配布されるデモンストレーション版ソフトウェアの多くは、実際、完全版の機能をすべて含んでいる。そこに埋め込まれるソフトウェア・プロテクト機構は、完全版を購入するまでは、完全版へのアクセスが阻止されるように設計されている。これには2つの理由がある。まず、ソフトウェア出版社は、デモンストレーション用のソフトウェアと完全版のソフトウェアという異なるソフトウェアの開発を避けたいという希望があるからであり、次に、ソフトウェア出版社は、ユーザに、デモンストレーション版を完全版に変換する手段を提供することによって、ソフトウェアの完全版を、再度ユーザに配布する手間を省けるからである。

【0005】

【発明が解決しようとする課題】これまでの説明から、このシステムは、市場拡大のための有効なツールであるが、一方、重大な不利益もある。このようなデモンストレーション媒体は、ソフトウェア海賊にとって、主要なソースになってしまうからである。ソフトウェアの海賊は、通常、ソフトウェア出版社が、そのソフトウェアにどのようなプロテクト機構を埋め込んだとしても、その機構を回避するように、デモンストレーション版を修正することができる。ソフトウェアのデモンストレーション版がそのように修正されてしまうと、配布されたプログラムに隠れていたプロテクト機構は、出版社が気付かないうちに解除されてしまう。さらに、いったん、ソフトウェア海賊が、ソフトウェア出版社が採用したプロテクト機構の回避に成功すると、いわゆる「クラック」と呼ばれる海賊版パッケージが作られ、他の人が同様に出版社に代金を支払わずに、デモンストレーション版のソフトウェア・プログラムを完全版に変換するための手助けをすることになる。この「クラック」プログラムが広範囲に渡って無料配布されると、ソフトウェア出版社は、多大な損害を受けることになる。

【0006】当分野において、デモンストレーション版のソフトウェア・プログラムを、海賊の攻撃から防衛するように設計されたプロテクト手段がいくつか知られている。このようなプロテクト技術は、典型的には、ソフトウェア・プログラムのバイナリ実行形式に対し、内部機能を付加することによって、デモンストレーション版に制限を加えている。このプロテクト技術によって、デモンストレーション版の制限を回避するように設計された非認可の修正版が検出され、その結果、例えば、特定

の画面が表示されるか、または、そのプログラムが自動的に実行されなくなることが好ましい。しかしながら、バイナリ実行プログラム（例えば、ウインドウズ™のオペレーティング・システムで「.EXE」の拡張子を有するファイル）は、通常、ソフトウェア・プログラム全体から見れば、その容量は非常に小さい。多くの消費者向けソフトウェア・アプリケーションにおいて、実行可能なファイルは、1メガバイトぐらいの大きさであるが、プログラムを実行するのに必要なデータ・ファイルおよびその他のファイルは、全体で、数百メガバイトを超える。このように、ソフトウェア・プログラムの中心的なバイナリ実行ファイルのみをプロテクトしても、実際には、デモンストレーション版ソフトウェア全体の非常に小さい部分しか海賊行為からプロテクトされないのである。

【0007】この技術によってプロテクトされたソフトウェア・プログラムが無料配布されると、最も単純な形式の「クラック（解読）」プログラムの場合には、プロテクトされた中心的なバイナリ実行ファイルを、合法的に購入されたプロテクトされていない正式版の実行ファイルに置き換えることができる。この「クラック」形式は、高度な技術を必要としないため、簡単に生成できる。これは、ソフトウェア・プロテクト技術における重大な弱点となっている。

【0008】他のソフトウェア・プロテクト機構として、暗号技術が使われている。暗号はコンピュータ・メッセージを一度だけ伝送するときには、十分に解決策となるが、ソフトウェア・プロテクト機構としては、概して、十分とは言えない。この技術においては、プロテクト・ソフトウェア・プログラムが走るたびにその鍵を用いなければならないので、暗号システムに必要な暗号キーは本質的に発見され易くなる。

【0009】以上のことから、改良されたソフトウェア・プロテクト機構が必要となる。

【0010】本発明の目的は、コンピュータ・ソフトウェア・プログラムに対する改良されたソフトウェア・プロテクト機構を提供することにある。本発明においては、ソフトウェア・プログラムの「アセット・ファイル」（すなわち、データ・ファイル）を変換し、コア実行プログラム自身を修正するシステムが開示される。この方法では、修正された実行プログラムを再配布するときは、それに対応する変換されたアセット・ファイルも再配布される。非変換（すなわち、オリジナル形式）のアセット・ファイルは修正された実行ファイルと一緒になければ（また、その逆も同様）、再配布は実現されない。

【0011】

【課題を解決するための手段】プロテクトされたソフトウェア・アプリケーションを配布できる形態にするために、そのアプリケーションは、まず、主要な構造要素に

分解される。すなわち、コア実行プログラムとアセット・ファイル（読み出し専用データ・ファイルおよび/または読み出し/書き込みデータ・ファイル）に分解される。変換キーを用いて、変換機能はアセット・ファイルに適用され、変換データ・ファイルを作る。その後、変換キーは、後に使用するためにストアされる。その後、プロテクト・モジュール・プログラムの実行用の呼を、コア実行プログラムに組み込むことによって、修正された実行プログラムが形成される。プロテクト・モジュール・プログラムは、前記の修正された実行プログラムを実行するときのみ、要求され、他の状態のもとでは動作しない。その後、ファイル変換モジュール・プログラムが生成され、大量のソフトウェアコンポーネントに付加される。変換キーと変換データ・ファイルのリストは、プロテクト・モジュール・プログラムに任意に付加される。その後、大量のソフトウェアコンポーネント（変換されたデータ・ファイル、変換キー、修正された実行プログラム、プロテクト・モジュール・プログラムおよびファイル変換モジュール・プログラム）がユーザへの配布用媒体に変換される。

【0012】ソフトウェア・プログラム・プロテクト・システム上のオペレーションにおいて、修正された実行プログラムがロードされる。このプログラムは、自動的に、このプロテクト・モジュール・プログラムを走らせる。このプロテクト・モジュールは、ファイル変換モジュールのオペレーションを起動し、修正された実行プログラムによってコンピュータ・オペレーション・システムになされた要求（例えば、ファイル読み出し、ファイル書き出し、ファイル・オープン、およびファイル・クローズなど）を妨害するように設計されている。ファイル変換モジュールは、変換キーを用いて、ファイル・オペレーション要求が変換データとして識別されたときだけ、要求された逆変換を実行する。

【0013】本発明は、プロテクト・モジュール・プログラムを使用することによって、修正された実行プログラムを、変換データ・ファイルに接続する。このように、ファイル変換モジュールは、どのような場合にも、修正された実行プログラムが存在するときおよびそのプログラムの制御下にあるとき以外は、要求された逆変換を実行する。

【0014】本発明の他の側面と特徴は、実施の形態および添付の図面を用いて説明することによって、当業者に明確になる。

【0015】

【発明の実施の形態】図1は、プロテクト機構を持たない典型的なインストールおよびランタイム構造を示すブロック図である。このようなソフトウェア・プログラムは、通常、CD-ROM10または他の物理的な媒体で配布される。本発明においては、配布ソースはCD-ROM以外（例えば、DVD、インターネットによるダウ



ンロードなど)でもよいが、便宜上、CD-ROMであると仮定する。

【0016】インストール・プログラム20は、CD-ROM10上で走り、バス11を介してCD-ROM10の内容を読み込み、それをユーザPCの内蔵ディスクと、コア実行プログラム30と、多数のファイルの組40、41、50および51に対してコピーする(バス21、22、23、24および25を介して)。コア実行プログラム30は、典型的には、コア実行プログラム41でサポートされる実行オペレーティング・システム・コードに依存する。このコア実行プログラム41は、更新されてもよいし、または、インストール・プログラム20によって初めからインストールされてもよい。または、ユーザのPCにすでにインストールされていてもよい。一例として、マイクロソフト社のウィンドウズ<sup>TM</sup>のDirectX<sup>TM</sup>グラフィック機能がある。コア実行プログラム30も、典型的には、コア実行サポート・ファイル40に依存し、また、インストール・プログラム20によってインストールされる。この例としては、ウィンドウズ<sup>TM</sup>のオペレーティング・システム中で特定のアプリケーションを用いる1以上のダイナミック・リンク・ライブラリ(DLL)がある。

【0017】適切に運転するために、コア実行プログラム30は、典型的には、データ・ファイルと相互作用し、また、データ・ファイルを操作する。例えば、読み出し/書き込みファイル・セット51は、ファイル・システム・インタフェース61を介してアクセスされるように示されている。また、読み出し専用ファイル50は、ファイル・システム・インタフェース60を介してアクセスされるように示されている。ファイル・セット50と51は、「アセット・ファイル」として知られている。図1には、ソフトウェア・プロテクト・アプリケーションは示されていないので、ソフトウェア・アプリケーションは、容易にコピーされ、配布される。

【0018】図2は、当分野で既知の典型的なソフトウェア・プロテクト機構を付加したソフトウェア・プログラムのインストールとランタイム構造を示すブロック図である。修正実行ファイル30Aとプロテクト・モジュール・プログラム31(これ以降、「プロテクト・モジュール31」と呼ぶ)を除いて、図2のすべてのコンポーネントは、図1に示すコンポーネントと同一である。コンポーネント30Aと31は、これらがプロテクトされていることを視覚的に分るように、破線で囲まれている。

【0019】図2の修正実行ファイル30Aは、ファイルの完全性をプロテクトするために、図1のオリジナルのコア実行プログラム30にいくらか修正を加えたものである。一般に、付加的なプロテクト・サポート・モジュール実行コンポーネント(「プロテクト・モジュール」)31は、パス26経由でインストールされ、修正

実行プログラム30Aの制御下において、ランタイム62で使用される。プロテクト・モジュール31は、必ずしも、プログラム実行中に使用可能ではないが、典型的には、プログラムがデモンストレーション・モードにある間、修正実行プログラム30Aを監視し、および/または修正実行プログラム30Aの機能をセットアップすることによって、予め設定された制限に合うようにする。

【0020】図2で示すシステムの弱点は、もし、プロテクト実行プログラム30Aがオリジナルのコア実行プログラム30のコピーによって置き換えられると、すべてのプロテクトが失われ、その結果、このアプリケーションは、非認可の完全機能版と同じになる。そのコンポーネントがまだ存在していたとしても、プロテクト・モジュール31によって提供されるプロテクト機構は失われてしまう。というのは、このようなプロテクト機構は、修正実行プログラム30Aによってのみ呼び出され、もし、プロテクトされていないコア実行プログラム30が修正実行プログラム30Aの代わりに使用されれば、起動されることはなく、休止状態のままである。

【0021】図3は、アセット・ファイル用ソフトウェア・プロテクト機構が追加された、ソフトウェア・プログラムのインストールおよびランタイム構造を示すブロック図である。図3において、コンポーネント20、40、41のみが、図1および図2で示すコンポーネントと同じである。CD-ROM10Aを除いて、残りのコンポーネントは破線の輪郭線で示される。これは、これらのコンポーネントが、本発明のソフトウェア・プロテクト機構に従ってプロテクトされていることを意味している。CD-ROM10Aは、プロテクトされたソフトウェア用の配布媒体である。図2で示すように、修正実行プログラム30Aは、ファイルの完全性をプロテクトするために、図1のオリジナルのコア実行プログラム30にいくらか修正を加えたものである。付加的なプロテクトサポート・モジュール実行コンポーネント(「プロテクト・モジュール」)31は、パス26経由でインストールされ、修正実行プログラム30Aの制御下で、ランタイム62で使用されるのは、図2と同じである。

【0022】図3において、図2のファイル・セット50と51は、変換ファイル・セット50Aと51Aによって置き換えられている。変換ファイル・セット50Aと51A中には、1以上の変換データ・ファイルがあってもよい。変換ファイル・セット50Aと51Aは、実行ファイルを含む任意のファイルのフォーマットを含んでもよい。変換ファイル・セット50Aと51Aは、インタフェース読み出し/書き込み機能61Aとインタフェース読み出し専用機能60Aを介して(また、図4に示される変換モジュール70を介して)、修正実行プログラム30Aによってアクセスされる。これらのインタフェース読み出し/書き込み機能60Aとインタフェー



ス読み出し専用機能61Aは、適切なランタイム変換と逆変換を提供する。変換と逆変換の正確な特徴は本発明の基本的な部分でないことは、当業者であれば理解できる。例えば、変換には、暗号化や、データ暗号化標準(DES)またはリベスト-シャミール-アルドレマン(RSA)のような標準アルゴリズムを用いた復号化、または他のもっと単純な暗号化および復号化アルゴリズムが含まれる。また、同じアプリケーション中で異なる変換の混合も使用される。唯一、実際に要求されることは、変換に可逆性があることであり、その変換が、ソフトウェア・プログラムの実行と変換の間で適切なバランスを提供することである。例えば、単純な変換では、秘密バイト値を用いて、ビット・シフトまたは排他的オア・オペレーションのようなバイトごとの論理オペレーションを変換バッファ・データに適用し、クリア・テキスト・データを作ってもよい。インタフェース機能60Aと61Aは、要求されたランタイム変換を提供し、種々の方法で実行される。例えば、要求された変換は、ソース・コードで組み込まれてもよく、または修正実行プログラム30Aに対するライブラリ・レベルで組み込まれてもよい。これらも、オペレーティング環境によって固有に提供されてもよく、または特別な媒体と関連してもよい。

【0023】図4は、本発明の一実施の形態のソフトウェア・プロテクト機構を示すブロック図である。図4は、変換ファイル・セット50Aと51A、修正実行プログラム30A、およびプロテクト・モジュール31を示している。これらはすべて、図3中の対応するコンポーネントと同様である。図3のコンポーネントのうちいくつかは、見やすくするために、図4では省略されているが、論理的には、図4のシナリオにおいても、同じように存在している。インタフェース読み出し専用機能60B、60Cおよび60D並びにインタフェース読み出し/書き込み機能61B、61Cおよび61Dは、本発明による適切なランタイム変換と逆変換を提供する。

【0024】本実施の形態において、インタフェース機能60B、60C、60D、61B、61Cおよび61Dは、修正実行プログラム30Aに対し透明である。すなわち、変換ファイル・セット50Aと51Aを用いる修正実行プログラム30A内のコードは、そのオリジナル・フォームのままである。なぜなら、本発明によれば、そのコードによって見られるファイルの入力/出力は、プロテクトされていない場合に見られるファイル入力/出力と同一であるからである。これは、オペレーティング・システム71とファイル変換モジュール・プログラム70(以下、「ファイル変換モジュール70」と呼ぶ)によってなされる。オペレーティング・システム71とファイル変換モジュール70は、インタフェース60B、60C、61B、61Cを介して、ファイル・オペレーションの読み出しおよび書き込みを仲介する。

オペレーティング・システム71(例えば、マイクロソフト・ウインドウズ95<sup>TM</sup>またはマイクロソフト・ウインドウズ98<sup>TM</sup>)およびファイル変換モジュール70は、そのファイル変換がファイル・オペレーションに透明に付加される手段である。

【0025】ファイル変換モジュール70を実行するには、種々の方法がある。しかしながら、本発明による本実施の形態によれば、このモジュールの目的は、(1)

修正実行プログラム30Aに対して、ファイル変換を完全に透明にすること、すなわち、ファイル変換の結果、このプログラムには何の修正も加えられないこと、

(11) 適切な環境下でのみ、この場合は、合法的な修正実行プログラム30Aに対してのみに、ファイル変換サービスを提供することの2点である。変換は、オリジナル・コア実行プログラム30のような他の実行プログラム、または関連のないプログラムに所属するファイルのような影響を受けないアセットに対して、実行されることはない。これらの目的を満たす手段は、以下に述べるように、ファイル変換モジュール70の論理によって初期化および実行される。

【0026】ファイル変換モジュール70のオペレーション論理は、図5、図6および図7のフローチャートに示される。オペレーションには明確な3つの位相がある。すなわち、初期化处理(図5)、ファイル・フック処理(図6)および終了処理(図7)である。

【0027】ファイル変換モジュール70は、オペレーティング・システム71からのファイル・アクティビティが妨害される前に初期化されなければならない。プロテクト・モジュール31は、修正実行プログラム30Aがあるときにだけ、アクティブである。例えば、ウインドウズ<sup>TM</sup>・オペレーティング・システムに対しては、プロテクト・モジュール31は、ダイナミック・リンク・ライブラリ(DLL)ファイルであってもよい。このDLLファイルは、修正実行プログラム30Aが生成される変換プロセスの一部として、オリジナルの実行プログラム30の「インポート・テーブル」に付加されてもよい。これは、修正実行プログラム30Aの実際のランタイム・バイナリ命令が実行される前に、修正実行プログラム30Aが最初に起動されたとき、自動的に、ウインドウズ<sup>TM</sup>・オペレーティング・システム中の「ローダ」の固有の動作を介して、プロテクト・モジュール31中のランニング特定コードの効果を待つようになる。他のプログラムで、プロテクト・モジュール31を走らせるものはない。プロテクト・モジュール31は、ここで論じられるように、呼を介して、変換モジュール70をセットアップするために、ファイル変換モジュール70は、他のすべてのプログラムに対してサービスを提供するためにセットアップされることはない。オリジナルのコア実行プログラム30は、プロテクト・モジュール31を起動しないばかりでなくそれがトリガする変換サ

ービスを起動しない。

【0028】図5のフローチャートで示すとおり、ステップ501と502で、プロテクト・モジュール31は、まずファイル変換モジュール70を初期化する。この段階で、情報は、プロテクト・モジュール31を介して、ファイル変換モジュール70に送られる。このような情報には、アセット・リスト（すなわち、変換ファイル・セット50Aと51Aのメンバー）、呼プロセス識別情報、変換キー、および変換アルゴリズムのような他の選択的情報が含まれる。呼プロセス情報は、オペレーティング・システム71から得られ、アセット関連情報は、プロテクト・モジュール31に埋め込まれるか、または、他のファイルを読み出すことによって、プロテクト・モジュール31から得てもよい。アセット関連情報も、他の手段によって、ファイル変換モジュール70から得てもよい。例えば、これは、ファイル変換モジュール70によって直接ファイルから読み出されるか、または、ファイル変換モジュール70自身に埋め込むこともできる。本実施の形態において、ステップ503で、プロテクト・モジュール31は、利用可能なオペレーティング・システム・サービスを用いて、ファイル変換モジュール70を、ファイル・システムに「フック（妨害）」する。例えば、ウインドウズ95<sup>TM</sup>またはウインドウズ98<sup>TM</sup>では、ファイル変換モジュール70は仮想装置ドライバ（VxD）であってもよく、プロテクト・モジュール31は、ウインドウズ<sup>TM</sup>の「仮想ファイル・システム」呼を用いて、そのオペレーティング・システムが、ファイル変換モジュール70を後続のファイル・システム呼上で実行されるオペレーションの低レベル・シーケンスの一部として起動するように調整してもよい。他の例として、13・ヘキサデシマル中断用の「終端および常駐」（TSR）プログラム中のソフトウェア中断ハンドラは、マイクロソフト・ディスク・オペレーティング・システム（MS-DOS<sup>TM</sup>）に対して使用することができる。図4はこのフック（妨害）機能を示し、この機能では、オペレーティング・システム71に対するインタフェース読み出し／書き込み機能61Bとインタフェース読み出し専用機能60Bは、インタフェース機能60Cと61Cで、ファイル変換モジュール70によって妨害される。本実施の形態において、このような妨害は、ファイル読み出し、ファイル書き込み、ファイル・オープン、またはファイル・クローズ・オペレーションが実行されるときはいつでも、行われる。初期化プロセスは、その後、ステップ553で完了する。フックが確立され、ファイル変換モジュール70が初期化されると、ファイル変換モジュール70は、後続のファイル要求によって自動的に起動される。

【0029】ファイル変換モジュール70は、いったん初期化されると、オペレーティング・システム71に送られたファイル・オペレーティング要求を受け取る。オ

ペレーティング・システム71への要求が、修正実行プログラム30Aへの、または修正実行プログラム30Aからの変換データに対してのものであると識別されるときにのみ、ファイル変換モジュール70は、要求された逆変換（すなわち、「変換サービス」）を実行する。もし、ファイル変換モジュール70が、その要求が修正実行プログラム30Aに向けられたものでもなく修正実行プログラム30Aからのものでもない判断すれば、妨害要求は、通常の処理をするために、オペレーティング・システム71に戻される。あらゆる場合において、ファイル変換モジュール70は、オペレーションに関するファイル・データ（もし、あれば）が、ファイル変換モジュール70に知られているメモリ・バッファ中でオペレーティング・システム71によってすでに置き換えられるように起動される。その後、ファイル変換モジュール70は、このバッファ・データに変換を行うオプションを有するか、または、単に、オペレーティング・システムに戻り、データを発見されたときの状態のままにしておく。

【0030】ファイル変換モジュール70に関するファイル・オペレーションのセットは、本発明の異なる実行に従って変化する。少なくとも、ファイル・オープン、ファイル読み出し、およびファイル・クローズ・オペレーションとは関連がある。ファイル書き込みオペレーションを含む他のオペレーションも、ファイル変換モジュール70を起動する同じオペレーティング・システム・フックを介して、トリガとして使用される。

【0031】これらのファイル・オペレーションに関するさらなる情報を以下に説明する。

(i) ファイル・オープン： マイクロソフト・DOS<sup>TM</sup>、マイクロソフト・ウインドウズ<sup>TM</sup> およびユニックス<sup>TM</sup> のようなオペレーティング・システム環境は、整数ファイル識別子の概念を使用して、オープン・ファイルを追跡する。プロセス・アセット・リスト（すなわち、変換ファイル・セット50Aと51A）にリストされているファイルに対し、安全プロセス（すなわち、修正実行プログラム30A）から発生したファイル・オープン要求によって、ファイル変換モジュール70は、関連変換キーによって記録されるべきオープン・ファイル用にオペレーティング・システム71によって指定された独自のファイル識別子を記録し、このファイルを、アクティブ・ファイル・リストに付加する。

(ii) ファイル読み出し： 上記のファイル・オープン中に確立された独自のファイル識別子を有するファイル読み出しオペレーションが、プロテクトされたアセット・ファイルの独自の識別子に対して起こるとき、オペレーティング・システム71から戻されたデータは、ファイル変換モジュール70によって復号化される。これは、オペレーティング・システム71に、呼プロセスによって特定された宛先バッファへの読み出しオペレ

ションを完了させることによって、また、引き続き、アセット・ファイルの関連キーによってデータを適切に（すなわち、同じ宛先バッファ中で）変換させることによって達成される。

（i i i）ファイル書き込み：図6には、ファイル書き込みオペレーションの処理は示されていないが、これはファイル・オープン・オペレーションに類似している。変換モジュール70は、オペレーティング・システム環境を呼び出す前に、呼プロセスのデータ・バッファ中で、データを適切に逆変換して、実際のファイル書き込みオペレーションを実行する。

（i v）ファイル・クローズ：変換モジュール70がそのファイルを閉じる要求を受け取ると、オープン・プロテクト・アセット・ファイル用のファイル識別子とキーをストアするために使用されるリソースは、破棄される。

【0032】図6は、本発明のファイル変換モジュール70で行われるオペレーションのステップを表わすフローチャートである。ステップ510で、ファイル変換モジュール70への「フック」は、オペレーティング・システム71によって起動される。ファイル変換モジュール70は、その要求が修正実行プログラム30Aへ（または、から）のものでないと判断すると、妨害要求は、通常の処理をするために、ステップ524と525で、オペレーティング・システム71に戻される。

【0033】しかしながら、呼プロセスがプロテクトされている（つまり、要求が、修正実行プログラム30Aへ（または、から）の変換データに対するものである）場合には、ファイル・オペレーションの解析は、ステップ512で行われる。図6において、実行すべきオペレーションは、ファイル・オープン、ファイル・クローズ、およびファイル読み出しのオペレーションである。「ファイル書き込み」オペレーションを含む他のオペレーションも、同様に、妨害することができる。

【0034】オペレーション・モードがファイル読み出しのとき、ステップ516で、ファイル識別子が、アクティブ・ファイル・リスト中にあるかどうか判断される。もしなければ、妨害された要求は、ステップ517とステップ525で、通常処理のため、オペレーティング・システム71に戻る。ファイル識別子が、アクティブ・ファイル・リスト中にあれば、ファイル変換モジュール70は、ステップ521で、オペレーティング・システム71に戻り、オペレーティング・システム71は、通常の方法で低レベル読み出しを完了する。これは、オペレーティング・システム71が、ステップ522で、ファイル変換モジュール70の制御に戻る（図示されていないが、ファイル変換モジュール70の論理を介して）ような方法で行われる。このアセットに関する正しい変換キーは、ステップ522で決定され、読み出しバッファ中のデータは、ステップ523で変換され

る。前記のデータは、「フック」された読み出しオペレーションの通常の方法で、オペレーティング・システム71によって、そこに置かれ、ファイル変換モジュール70を起動する。この処理は、その後、ステップ525で完了する。

【0035】オペレーション・モードがファイル・クローズのとき、ステップ518で、ファイル識別子がアクティブ・ファイル・リスト中にあるかどうか判断される。もしなければ、妨害された要求は、ステップ517とステップ525で、通常処理のためにオペレーティング・システム71に戻る。ファイル識別子がアクティブ・ファイル・リスト中にあれば、ステップ519で、ファイル識別子と変換キーは、アクティブ・ファイル・リストから除去される。ステップ520で、ファイル・クローズ・オペレーションが実行される。この処理は、その後、ステップ525で完了する。

【0036】図7は、本発明の終了処理ステップのフローチャートである。この終了ステップにおいては、ステップ551で、呼プロセスに割り当てられたすべてのリソースが解放され、ステップ552で、オペレーティング・システム環境フック機構が除去される。その後、終了処理ステップはステップ553で完了する。

【0037】図8は、本発明の方法に従って、ソフトウェア・プログラムがアセットを変換するプロセスを示すブロック図である。これは、ソフトウェア・プログラムを配布状態に変換する前段階の配布変換プロセスである（図4参照）。まず、ソフトウェア・プログラムのオリジナルの配布形態10は、プロセス100によって、キーコンポーネント（すなわち、インストール・プログラム20、読み出し専用ファイル・セット50、読み出し/書き込みファイル・セット51、コア実行サポート・ファイル40、一般実行システム・リソース41、およびオリジナル・コア実行プログラム30）に分解される。

【0038】その後、ソフトウェア変換プログラム101が実行される、この論理は、図9のフローチャートに示される。このソフトウェア・プログラムは、典型的には、ユーザ入力によって、ランタイム非実行コンポーネントである読み出し専用ファイル・セット50と読み出し/書き込みファイル・セット51から、特定のセット・プログラムを選択する。ソフトウェア変換プログラム101は、選択されたコアセット・ファイルごとに、変換アルゴリズムとキーを選択する。その後、ファイルごとに選択された暗号化アルゴリズムは、その特定のファイル用のキーを使用して適用される。

【0039】オリジナル・コア実行プログラム30も、修正が必要である。特に、オリジナル・コア実行プログラム30は、プロテクト・モジュール31を起動する修正実行プログラム30Aに変換される。例えば、ウインドウズ<sup>TM</sup>環境では、これは、オリジナル・コア

10

20

30

40

50

実行プログラム30の入力テーブルの拡張版中のDLLに、参照を付加することによって達成される。プロテクト・モジュール31およびファイル変換モジュール70は、ファイル・セットに付加されなければならない。これらは、個別のファイルとしてストアされるか、または、プロテクトの理由から、ファイル30A、31および/または70のような他のファイルに組み込まれる。

【0040】最終的には、非ランタイム・コンポーネントにも、修正が必要となる。特に、ソフトウェア・プログラムのインストール・プログラム20は、上述の新しいコンポーネントおよび変化したコンポーネントを含むように強化される必要があり、その結果、修正インストール・プログラム20Aとなる。

【0041】このプロセスが終了すると、ランタイム・コンポーネントの新しいセット（すなわち、修正実行プログラム30A、プロテクト・モジュール31、ファイル変換モジュール70、および変換ファイル・セット50Aと51A）が存在するようになる。これらのコンポーネントは、その後、ソフトウェア・ユーザに配布するために、プロセス102を用いて、CD-ROMまたは他の媒体に変換される。これらのコンポーネントは、それ以後、アプリケーション・ランタイムで必要に応じて起動される。

【0042】本発明にとって、前述の修正が、すべて、ソフトウェア変換プログラム101によって実行されるかどうか、または、オリジナル・コア実行プログラム30および/またはインストール・プログラム20上で実行されるような変換が、独立ではあるが関連したプロセスによって実行されるかどうかは重要ではない。また、厳密には、修正実行プログラム30Aがファイル変換モジュール70を（通常は、プロテクト・モジュール31を介して）起動するかぎり、他の変換のどのような性質を用いて修正実行プログラム30Aを生成するかも重要ではない。

【0043】コア実行プログラム30を、修正実行プログラム30Aに変換するには、他の方法もある。これらの方法も、ファイル変換モジュール70の変換を有効に制御できる。例えば、修正実行プログラム30Aを生成する変換プロセスは、非実行データを、コア実行プログラム30に付加する。この結果、このデータは、ファイル変換モジュール70によって、ランタイムで、発見され検査される。このデータは、「ライセンス」の形式として、また、この特定のプログラムに変換を加えるかどうかを決定する際に、ファイル変換モジュール70によって使用される存在およびコンテンツとして機能する。別の変換機能として、この変換プロセスは、コア実行プログラム30に、他の実行「コールバック」機能を付加することができる。この機能は、前に存在するプログラム30の実行コードには何の関係も必要としないが、前記のモジュールが、変換サービスを提供するかど

うかを判断するとき、ファイル変換モジュール70によって呼ばれる。前述した種々の実施の形態においては、ファイル変換モジュール70の最初の初期化は、コア実行プログラム30によって（または、もしあれば、プロテクト・モジュール31によって）起動されることはないが、アプリケーション・インストール・プログラム20のようないくつかの代替手段によって初期化される。

【0044】図9は、図8で示すファイル変換プロセスのフローチャートである。ステップ601で、ファイル変換プロセスは開始される。ステップ602で、ユーザは、新しいランタイムおよび非ランタイム・コンポーネントの新たなセット（すなわち、修正実行プログラム30A、プロテクト・モジュール31、ファイル変換モジュール70、変換ファイル・セット50Aと51A、および修正インストール・プログラム20A）に対する宛先ディレクトリを選択する。ステップ603で、ユーザは、本発明に係る変換されるべきコア実行ファイル（実行可能な）を選択する。ステップ604で、ユーザは、また、変換されるべきデータ・ファイル（「アセット・ファイル」としても知られている）を選択する。ステップ605で、ソフトウェア変換プログラムは、変換キーを生成する。ステップ606で、選択されたアセット・ファイルは、変換キーを使用して変換され、宛先ディレクトリ中に置かれる。決定ステップ607を使用して、ステップ604から606は、これ以上、変換するべきアセットがなくなるまで、繰り返される。この点において、アセット・リストと関連変換キー・セットが使用され、他のファイルに別々にストアされ、または組み込まれる。

【0045】ステップ608で、プロテクト・モジュール31が検索され、前述のアセット・ファイルと変換キー情報が選択的に付加される。ステップ609で、修正実行プログラム30Aは、プロテクト・モジュール31に依存して生成される。前述のように、そのような依存は、ウインドウズ<sup>TM</sup>の環境では、典型的には、プロテクトDLLを参照する修正実行プログラム30Aに、インポート・テーブル参照を付加することである。この発明の一実施の形態においては、オリジナル・コア実行プログラム30をプロテクト・モジュール31にリンクすることは、オリジナル・コア実行プログラム30になされる唯一の修正であり、それによって修正実行プログラム30Aを生成する。ファイル変換プロセスは、ステップ610で完了する。

【0046】以下は、本発明の一実施の形態によるオペレーションの一例である。

【0047】1. ソフトウェアの出版元は、ウインドウズ95環境用のソフトウェア・プログラムを作っている。この一例として、種々のレベルを備えたロールプレイ・エンターテインメント・プログラムがある。この例

では、各レベルは、50メガバイトの読み出し専用データ・ファイル（アセット）で示される。コア実行ファイルは、2メガバイトのEXEファイルである。

【0048】2. 本発明では、図8に示すように、各データ・ファイル50が変換データ・ファイルに変換され、これらの変換データ・ファイルに依存して、修正実行プログラム30Aが生成されるように、ソフトウェア・プログラムが変換される。

【0049】3. このソフトウェア・プログラムの変換形式は、パッケージされて、CD-ROMまたは他の媒体によってユーザに配布される。

【0050】4. ユーザは、ソフトウェア・プログラムをインストールし、走らせる。

【0051】5. インストールの際、修正実行プログラム30A、修正読み出し専用ファイル・セット50A、プロテクト・モジュール31およびファイル変換モジュール70は、すべてコピーされるか、または、ユーザのコンピュータで使用される。これは、ソフトウェア・プログラムに対して通常使用されるすべてのコンポーネントとは別なものである。プロテクト・モジュール31は、オブジェクト・リンクおよび組み込み（OLE）オブジェクト、個別の実行ファイル、またはダイナミック・リンク・ライブラリ（DLL）ファイルの形式を取る。ウインドウズ95<sup>TM</sup>、またはウインドウズ98<sup>TM</sup>において実行可能なファイルの場合、ファイル変換モジュール70としては、ウインドウズ<sup>TM</sup>仮想デバイス・ドライバ（VxD）が最適である。本実施例においては、ソフトウェア・プログラムの読み出し/書き込みファイルは影響されることはない。

【0052】6. プログラムが走ると、まず、ユーザに第1レベルのソフトウェア・プログラムが供給される。これは、第1のレベルのデータ・ファイルが、必ず、修正実行プログラム30Aによって、アクセスされ読まなければならないことを意味する。図6のフローチャートによれば、最初のファイル・オープン・オペレーションは、ファイル変換モジュール70をトリガする。このファイル変換モジュール70は、ファイルとファイル読み出しプログラムの特定の組み合わせで変換が行われることを決定する。

【0053】7. レベル1のデータ・ファイルに対するそれ以降のすべてのファイル読み出しは、ファイル変換モジュール70を起動する。図6によると、ファイル変換モジュール70は、ファイル・システムに、実際の読み出しを行わせ、バッファ・ファイル・データを、使用される前に適切に変換し、オペレーションシステム71によって、修正実行プログラム30Aに変換する。このことは読み出し/書き込みファイルに対しては一般的ではないが、書き込みは、同様に妨害され、必要な場合には変換される。

【0054】8. プログラムが終了するか、または、ユ

ーザがソフトウェア・プログラム内の他のレベルに移動すると、レベル1のファイルは閉じる。そのとき、変換モジュール70は、そのデータ・ファイルに関連するすべての情報をリセットする。本発明によれば、そのファイルのそれ以降のすべての操作は、ローカル・ファイル・システムと変換モジュール70の両方によって要求された時に、ファイル・オープン・オペレーションによって開始される。

【0055】今まで、本発明の好適な実施の形態について説明してきたが、本発明の精神から逸脱することなく変形および改良ができるので、本発明は限定的に解釈されるべきではない。本発明の範囲は、以下の請求の範囲とそれに関連する明細書の記載によって定義される。

【図面の簡単な説明】

【図1】 プロテクト機構を持たないソフトウェア・プログラムの典型的なインストールとランタイムを示すブロック図である。

【図2】 典型的なソフトウェア・プロテクト機構を付加したソフトウェア・プログラムのインストールとランタイムを示すブロック図である。

【図3】 アセット・ファイル用のソフトウェア・プロテクト機構を付加したソフトウェア・プログラムのインストールとランタイムを示すブロック図である。

【図4】 本発明の一実施の形態によるソフトウェア・プロテクト機構を示すブロック図である。

【図5】 本発明の初期化ステップのフローチャートである。

【図6】 本発明の変換モジュールによるオペレーション・ステップのフローチャートである。

【図7】 本発明の終了ステップのフローチャートである。

【図8】 本発明の方法により、ソフトウェア・プログラムがアセットを変換するプロセスを示すブロック図である。

【図9】 図8で示されるファイル変換プロセスのフローチャートである。

【符号の説明】

10、10A…CD-ROM、

11…バス

20…インストール・プログラム、

20A…修正インストール・プログラム、

21～26…バス

30…コア実行ファイル（または、オリジナル・コア実行プログラム）、

30A…修正実行ファイル（または、修正コア実行プログラム、プロテクトプログラム）、

31…プロテクト・モジュール・プログラム（または、プロテクト・モジュール）、

40…コア実行サポート・ファイル、

41…実行オペレーティング・システム・コード（また

23

は、コア実行プログラム、一般実行システム・リソース)、

50…読み出し専用ファイル・セット(または、データ・ファイル)、

50A…変換ファイル・セット(または、修正読み出し専用ファイル・セット)、

51…読み出し/書き込みファイル・セット、

51A…変換ファイル・セット、

60, 61…ファイル・システム・インタフェース、 \*

24

\* 60A, 60B, 60Cおよび60D…インタフェース読み出し専用機能、

61A, 61B, 61Cおよび61D…インタフェース読み出し/書き込み機能、

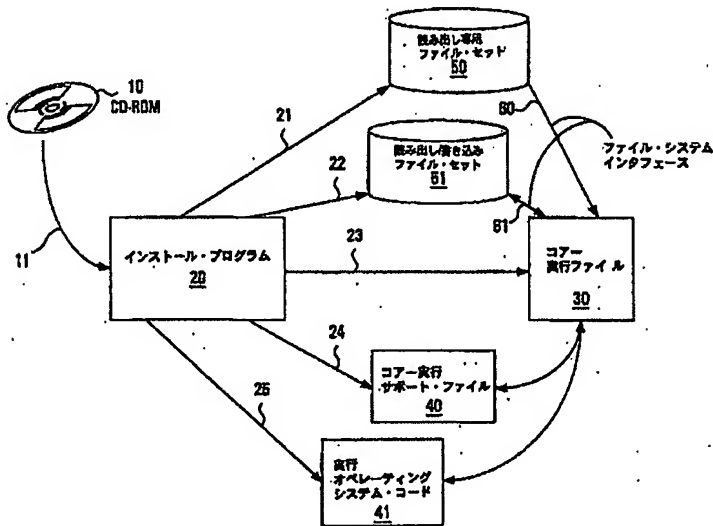
70…ファイル変換モジュール・プログラム、

71…オペレーティング・システム、

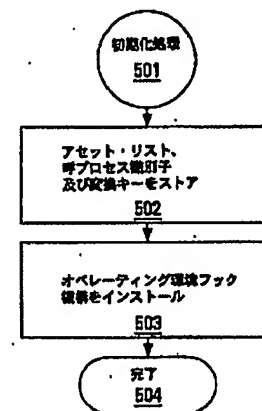
100…プロセス

101…ソフトウェア変換プログラム

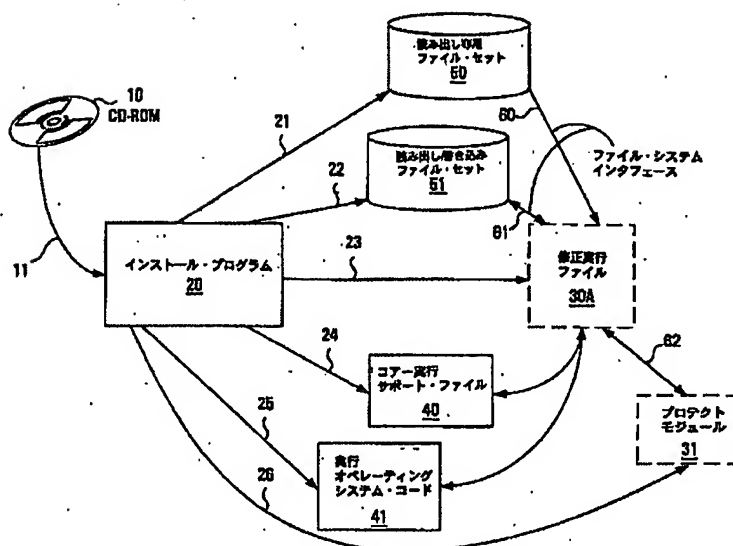
【図1】



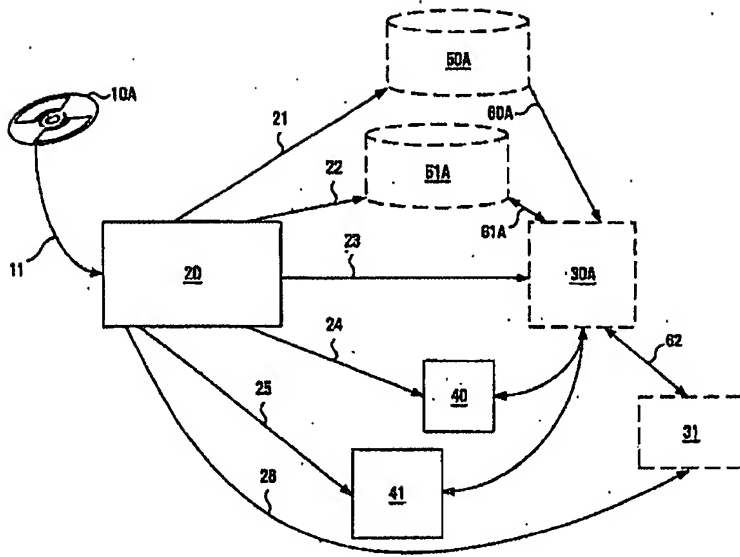
【図5】



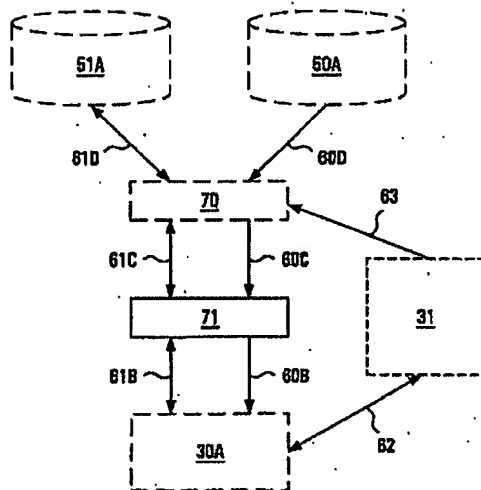
【図2】



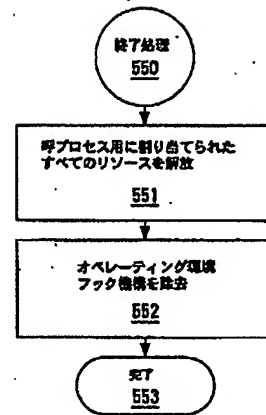
【図3】



【図4】

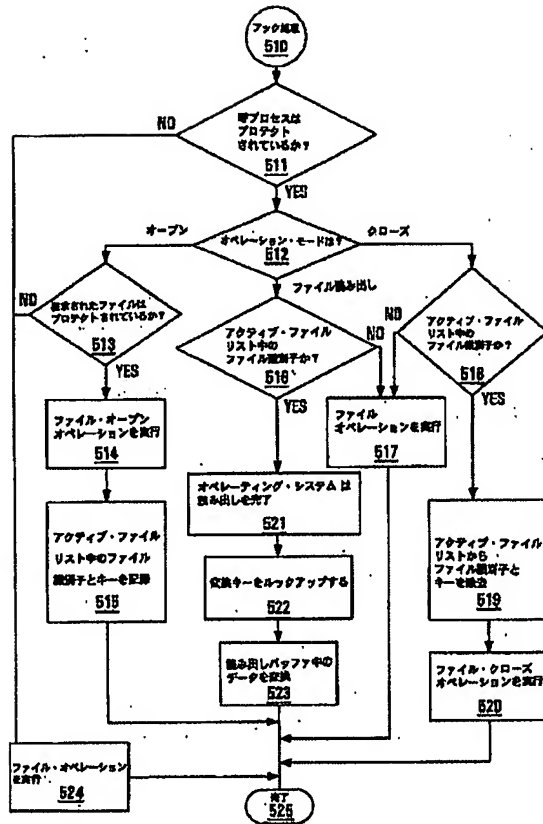


【図7】

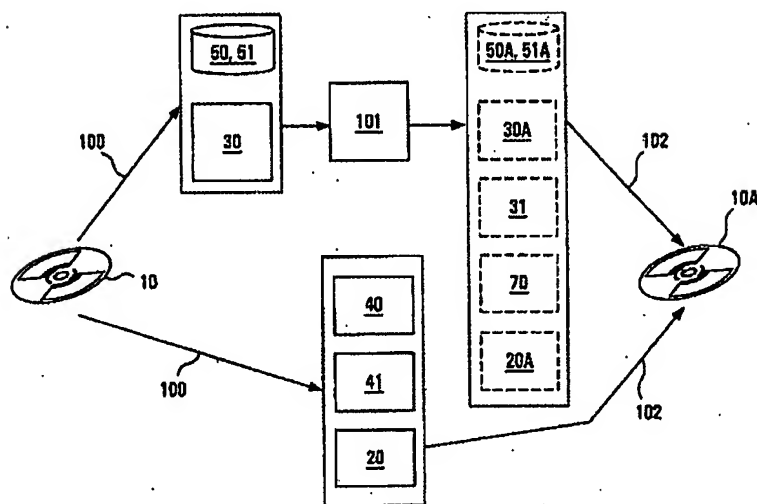




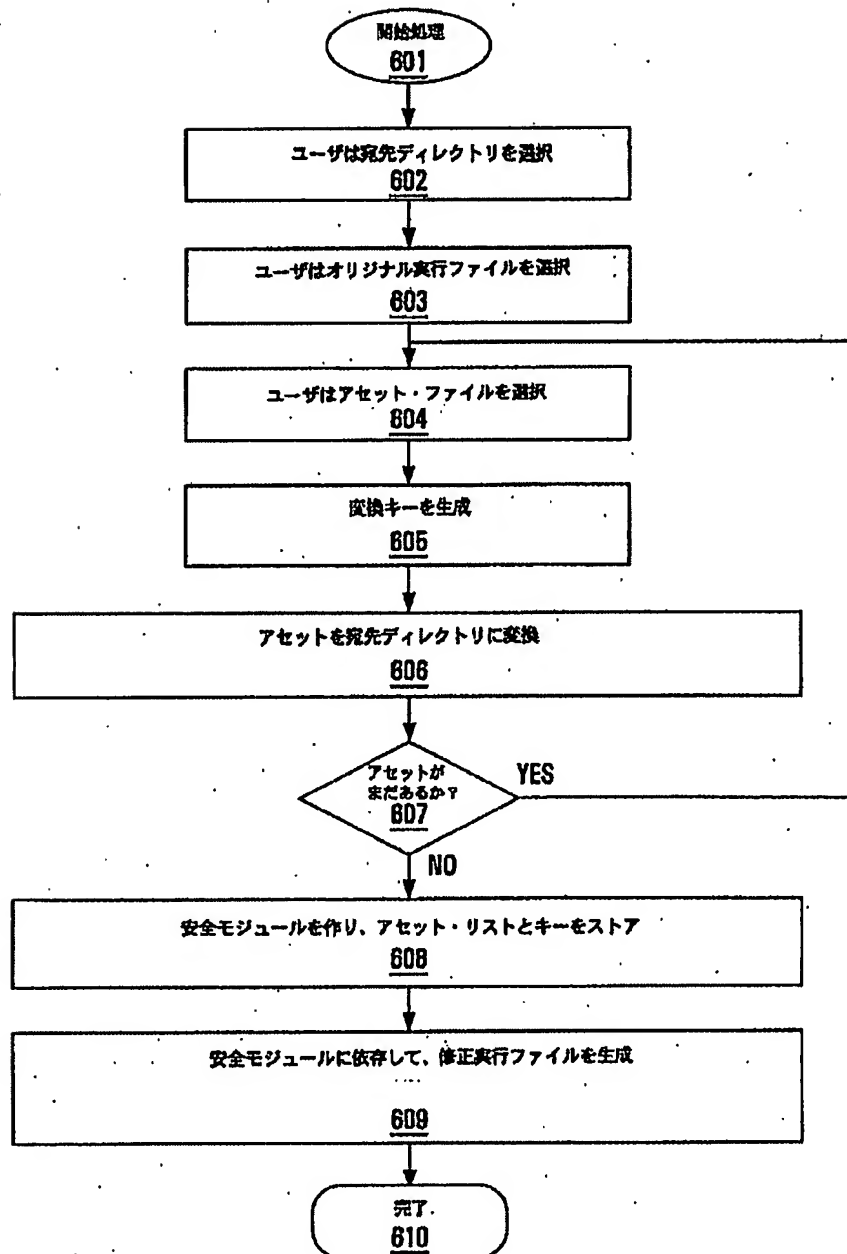
【図6】



【図8】



【図9】



フロントページの続き

(72)発明者 スコット・アラン・トムソン  
カナダ国、ケイ2エイ 1ティー9、オン  
タリオ、オタワ、エディソン アベニュー